W81

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 11-259487

(43)Date of publication of application : 24.09.1999

(51)Int.Cl. G06F 17/30

(21)Application number : 10-055560

(22)Date of filing : 06.03.1998

(71)Applicant : TOSHIBA CORP
TOSHIBA COMPUT ENG CORP

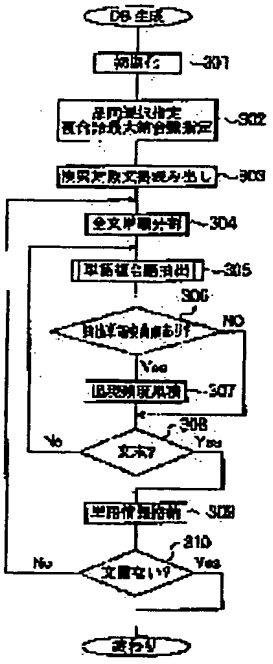
(72)Inventor : TANOSAKI YASUO
NISHINA TAKUYA
NAKAMOTO YUKIO
KUBOTA NAOHIDE

(54) SIMILAR DOCUMENT RETRIEVING DEVICE, SIMILAR DOCUMENT RETRIEVING METHOD AND STORAGE MEDIUM RECORDED WITH PROGRAM FOR RETRIEVING SIMILAR DOCUMENT

(57)Abstract:

PROBLEM TO BE SOLVED: To enhance retrieving accuracy when a similar document is retrieved by extracting a composite word from a retrieving key document and a document to be retrieved.

SOLUTION: When the composite word, i.e., 'brush character address printing function' exists in the retrieving key document or the document to be retrieved and the maximum number of connections is specified as three, all the composite words like 'brush character address', 'character address printing', 'address printing function', 'brush character', 'character address', etc., consisting of the number of words to be equal to or less than the maximum number of connections are extracted from each document, appearance frequency of the composite words is calculated and a degree of similarity between the retrieving key document and the document to be retrieved is calculated. Since different composite words to characterize the documents with specified contents are thoroughly extracted, a more proper degree of similarity between the documents is calculated and highly accurate retrieval of similar document intended by a user is performed.



LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]
[Date of registration]
[Number of appeal against examiner's decision of rejection]
[Date of requesting appeal against examiner's decision of rejection]
[Date of extinction of right]

W81

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平11－259487

(43)公開日　平成11年(1999)９月24日

(51)Int.Cl.[6]	識別記号	ＦＩ		
Ｇ０６Ｆ　17/30		Ｇ０６Ｆ	15/403	３３０Ｂ
			15/401	３１０Ａ
			15/403	３５０Ｃ

審査請求　未請求　請求項の数５　ＯＬ　（全 12 頁）

(21)出願番号　特願平10－55560

(22)出願日　平成10年(1998)３月６日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地

(71)出願人　000221052
東芝コンピュータエンジニアリング株式会社
東京都青梅市新町３丁目３番地の１

(72)発明者　田野崎　康雄
東京都青梅市末広町２丁目９番地　株式会社東芝青梅工場内

(74)代理人　弁理士　須山　佐一

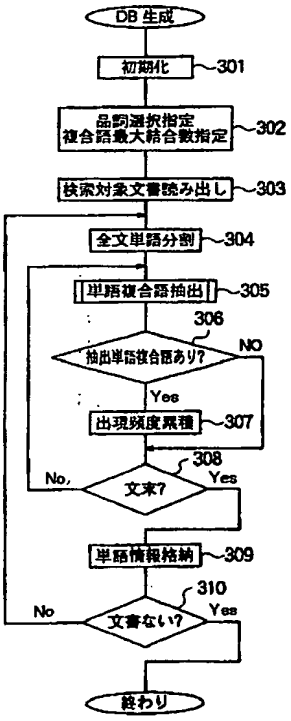
最終頁に続く

(54)【発明の名称】　類似文書検索装置、類似文書検索方法、および類似文書検索のためのプログラムが記録された記録媒体

(57)【要約】

【課題】　検索キー文書と検索対象文書から複合語を抽出して類似文書を検索する場合の検索精度の向上を図る。

【解決手段】　検索キー文書或いは検索対象文書に例えば「筆文字宛名印刷機能」といった複合語が存在し、最大結合数を３と指定したとき「筆文字宛名」「文字宛名印刷」「宛名印刷機能」「筆文字」「文字宛名」などの最大結合数以下の単語数からなる複合語を各文書からすべて抽出し、これらの複合語の出現頻度を計算して、検索キー文書と検索対象文書との類似度を算出する。特定の内容の文書を特徴付ける異なる複合語を漏れなく抽出することができるので、文書間のより妥当な類似度を計算でき、ユーザの意図する精度の高い類似文書検索を行うことができる。



【特許請求の範囲】

【請求項１】　ある文書を検索キー文書としてこの検索キー文書と類似する文書を複数の検索対象文書の中から検索する類似文書検索装置において、
前記検索キー文書および前記検索対象文書を単語単位に分割する分割手段と、
前記分割手段によって分割された単語の中から予め指定された条件を満たす単語を抽出する単語抽出手段と、
前記検索キー文書および前記検索対象文書から抽出すべき複合語を構成する単語数の上限値を指定する指定手段と、
前記単語抽出手段によって抽出された単語の結合により構成される複合語のうち前記指定手段により指定された上限値以下の数の単語により構成されるすべての複合語を前記検索キー文書および前記検索対象文書から抽出する複合語抽出手段と、
前記複合語抽出手段によって抽出された複合語の前記検索キー文書および前記検索対象文書での出現頻度をそれぞれ算出する手段とを具備することを特徴とする類似文書検索装置。

【請求項２】　ある文書を検索キー文書としてこの検索キー文書と類似する文書を複数の検索対象文書の中から検索する類似文書検索装置において、
前記検索キー文書および前記検索対象文書を単語単位に分割する分割手段と、
前記分割手段によって分割された単語の中から予め指定された条件を満たす単語を抽出する単語抽出手段と、
前記検索キー文書および前記検索対象文書から抽出すべき複合語を構成する単語の数の上限値を指定する指定手段と、
前記単語抽出手段によって抽出された単語の結合により構成される複合語のうち前記指定手段により指定された上限値以下の数の単語により構成されるすべての複合語を前記検索キー文書および前記検索対象文書から抽出する複合語抽出手段と、
任意の単語を不要語として選択する不要語選択手段と、
前記複合語抽出手段によって抽出された複合語のうち前記不要語選択手段によって選択された不要語を含む複合語を無効とする複合語無効化手段と、
前記複合語抽出手段によって抽出された有効な複合語の前記検索キー文書および前記検索対象文書での出現頻度をそれぞれ算出する手段とを具備することを特徴とする類似文書検索装置。

【請求項３】　請求項１または２記載の類似文書検索装置において、
前記単語抽出手段による単語の抽出条件として単語の品詞を指定する手段を有することを特徴とする類似文書検索装置。

【請求項４】　ある文書を検索キー文書としてこの検索キー文書と類似する文書を複数の検索対象文書の中から検索する類似文書検索方法において、
前記検索キー文書および前記検索対象文書を単語単位に分割し、
前記分割された単語の中から予め指定された条件を満たす単語を抽出し、
前記抽出された単語の結合により構成される複合語のうち予め指定された数以下の単語で構成されるすべての複合語を前記検索キー文書および前記検索対象文書から抽出し、
前記抽出された複合語の前記検索キー文書および前記検索対象文書での出現頻度を算出することを特徴とする類似文書検索方法。

【請求項５】　ある文書を検索キー文書としてこの検索キー文書と類似する文書を複数の検索対象文書の中から検索するためのプログラムが記録された記録媒体であって、
前記検索キー文書および前記検索対象文書を単語単位に分割する分割手段と、
前記分割手段によって分割された単語の中から予め指定された条件を満たす単語を抽出する単語抽出手段と、
前記検索キー文書および前記検索対象文書から抽出すべき複合語を構成する単語数の上限値を指定する指定手段と、
前記単語抽出手段によって抽出された単語の結合により構成される複合語のうち前記指定手段により指定された上限値以下の単語数で構成されるすべての複合語を前記検索キー文書および前記検索対象文書から抽出する複合語抽出手段と、
前記複合語抽出手段によって抽出された複合語の前記検索キー文書および前記検索対象文書での出現頻度をそれぞれ算出する手段とを具備するプログラムが記録されていることを特徴とする記録媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、文書データベースから、文書間の類似度に基づく文書データの検索を行う類似文書検索装置、類似文書検索方法、および類似文書検索のためのプログラムが記録された記録媒体に関する。

【０００２】

【従来の技術】近年、大量の電子化された文書データが流通するようになり、自動分類等を行う目的で、文書データベース中から指定された文書（以下、検索キー文書と呼ぶ）に類似する文書の自動検索を行うシステムが実用化されてきている。この文書検索システムでは、検索キー文書に含まれている単語と検索対象となる文書（以下、検索対象文書と呼ぶ）に含まれている単語とを比較し、共通する単語の種類、出現場所、出現回数などから空間ベクトル法により類似度を算出して、類似度の高い検索対象文書を検索結果として出力する。

【０００３】このような類似文書検索では、検索キー文書や検索対象文書から、その文書を特徴付ける単語を抽出することが、精度の高い類似文書検索を行うために非常に重要な鍵となる。これまで、文書から名詞やサ変名詞などの単語を対象に単語の抽出を行っていたが、文書から抽出された単一の単語が必ずしもその文書を特徴付ける単語として使用されているとは限らない。

【０００４】そこで、文書を特徴付ける複数の単語の結合からなる複合語を文書から抽出する方法が提案されている。このように文書を特徴付ける複合語を抽出することで、単位の単語を抽出する方法に比べ、ユーザの意図する検索結果をより高い精度で得ることができる。

【０００５】

【発明が解決しようとする課題】ところで、このように複合語を抽出する方法では、複合語の語長（単語数）が長くなればなるほど、その複合語がその文書を特徴付ける度合も高くなるが、その反面、検索対象の文書から抽出される単語の数が極端に減ってしまう傾向がある。このため、本来類似文書として検索されるべき文書に対する類似度として妥当な値が得られず、やはりユーザの意図する検索結果を高精度に得ることは困難であった。

【０００６】本発明は、このような課題を解決するためのもので、検索キー文書と検索対象文書から複合語を抽出して類似文書を検索する場合の検索精度の向上を図ることのできる類似文書検索装置、類似文書検索方法、および類似文書検索のためのプログラムが記録された記録媒体の提供を目的とする。

【０００７】

【課題を解決するための手段】上記した目的を達成するために、本発明は、請求項１に記載されるように、ある文書を検索キー文書としてこの検索キー文書と類似する文書を複数の検索対象文書の中から検索する類似文書検索装置において、前記検索キー文書および前記検索対象文書を単語単位に分割する分割手段と、前記分割手段によって分割された単語の中から予め指定された条件を満たす単語を抽出する単語抽出手段と、前記検索キー文書および前記検索対象文書から抽出すべき複合語を構成する単語数の上限値を指定する指定手段と、前記単語抽出手段によって抽出された単語の結合により構成される複合語のうち前記指定手段により指定された上限値以下の数の単語により構成されるすべての複合語を前記検索キー文書および前記検索対象文書から抽出する複合語抽出手段と、前記複合語抽出手段によって抽出された複合語の前記検索キー文書および前記検索対象文書での出現頻度をそれぞれ算出する手段とを具備することを特徴とする。

【０００８】本発明の類似文書検索装置では、文書から抽出された単語の結合により構成される複合語のうち、指定された上限値以下の数の単語により構成されるすべての複合語を検索キー文書および検索対象文書から抽出することで、文書を特徴付ける複合語を漏れなく抽出することができ、検索キー文書と検索対象文書との類似度をより高精度に算出することができる。

【０００９】また、本発明は、請求項２に記載されるように、ある文書を検索キー文書としてこの検索キー文書と類似する文書を複数の検索対象文書の中から検索する類似文書検索装置において、前記検索キー文書および前記検索対象文書を単語単位に分割する分割手段と、前記分割手段によって分割された単語の中から予め指定された条件を満たす単語を抽出する単語抽出手段と、前記検索キー文書および前記検索対象文書から抽出すべき複合語を構成する単語の数の上限値を指定する指定手段と、前記単語抽出手段によって抽出された単語の結合により構成される複合語のうち前記指定手段により指定された上限値以下の数の単語により構成されるすべての複合語を前記検索キー文書および前記検索対象文書から抽出する複合語抽出手段と、任意の単語を不要語として選択する不要語選択手段と、前記複合語抽出手段によって抽出された複合語のうち前記不要語選択手段によって選択された不要語を含む複合語を無効とする複合語無効化手段と、前記複合語抽出手段によって抽出された有効な複合語の前記検索キー文書および前記検索対象文書での出現頻度をそれぞれ算出する手段とを具備することを特徴とする。

【００１０】本発明の類似文書検索装置では、文書を特徴付ける複合語を漏れなく抽出することができ、検索キー文書と検索対象文書との類似度をより高精度に算出することができるとともに、予め指定された不要語の単語を含む複合語を無効なものとすることで、ユーザの意図をさらに反映したより一層高精度な類似文書検索を行うことが可能になる。

【００１１】

【発明の実施の形態】以下、図面を参照して本発明の実施形態を説明する。

【００１２】図１は本発明の実施形態である類似文書検索装置のハードウェア構成を示すブロック図である。

【００１３】同図に示すように、この類似文書検索装置は、キーボードなどの入力装置１、ＣＰＵおよびメモリなどから構成される制御装置２、類似文書の検索結果などを表示する表示装置３、および、文書データや類似文書検索のための各文書の単語情報や品詞辞書、不要語辞書などを格納する外部記憶装置４などから構成されている。外部記憶装置４に格納された品詞辞書、不要語辞書の構成を図６、図７にそれぞれ示す。

【００１４】図２にこの類似文書検索装置における制御装置１の構成を示す。制御装置１は、制御部１００とメモリ部２００から構成される。

【００１５】制御部１００は、初期化部１０１、入力部１０２、出力部１０３、抽出対象品詞設定部１０４、最大結合数設定部１０５、検索対象文書読み出し部１０

６、検索対象単語切り出し部１０７、検索対象単語複合語抽出部１０８、検索対象単語出現頻度算出部１０９、検索対象単語情報書き込み部１１０、検索キー文書入力部１１１、検索キー単語切り出し部１１２、検索キー単語複合語抽出部１１３、検索キー単語出現頻度算出部１１４、検索対象単語情報読み出し部１１５、共通単語抽出部１１６、類似度算出部１１７、検索結果出力部１１８などから構成される。メモリ部２００は、品詞情報バッファ部２０１、選択品詞情報バッファ部２０２、不要語情報バッファ部２０３、最大結合数バッファ部２０４、検索対象文書格納バッファ部２０５、検索対象全文分割単語格納バッファ部２０６、検索対象複合語候補格納バッファ部２０７、検索対象抽出複合語格納バッファ部２０８、検索対象単語情報格納バッファ部２０９、検索キー文書格納バッファ部２１０、検索キー全文分割単語格納バッファ部２１１、検索キー複合語候補格納バッファ部２１２、検索キー抽出複合語格納バッファ部２１３、検索キー単語情報格納バッファ部２１４、共通単語情報格納バッファ部２１５、算出類似度格納バッファ部２１６、検索結果出力バッファ部２１７などから構成される。

【００１６】初期化部１０１は、上記各バッファ部の初期化を行い、更に、外部記憶装置４における辞書（品詞辞書、不要語辞書など）の内容をメモリ部に読み込む。

【００１７】入力部１０２は、ユーザによる入力装置１からの検索キー文書や単語抽出条件の設定など各種設定の入力を行う。

【００１８】出力部１０３は、入力部１０２により入力された検索キー文書などの各種設定内容を表示装置３に出力する。

【００１９】抽出対象品詞設定部１０４は、ユーザが品詞情報バッファ部２０１から選択した抽出対象単語の品詞を選択品詞情報バッファ部２０２に格納する。

【００２０】最大結合数設定部１０４は、ユーザが指定した複合語の最大結合数を最大結合数バッファ部２０４に格納する。

【００２１】検索対象文書読み出し部１０６は、外部記憶装置４に格納されている検索対象文書に関する情報を文書データベース化するために、文書データベース化すべき文書情報を外部記憶装置４から読み込み、検索対象文書格納バッファ部２０５に格納する。

【００２２】検索対象単語切り出し部１０７は、検索対象文書格納バッファ部２０５に格納されている検索対象文書からの単語切り出しを行う。そして、その検索対象文書から抽出される全ての単語とその品詞を検索対象文書全文分割単語格納バッファ部２０６に格納する。単語の切り出しは形態素解析などにより行い、その文書から抽出される単語の品詞情報を「名詞」、「サ変名詞」、「形容詞」などで表現する。

【００２３】検索対象単語複合語抽出部１０８は、検索対象全文分割単語格納バッファ部２０６に格納されている全ての単語とその品詞情報の中から、選択品詞情報バッファ部２０２に格納されている品詞情報を参照して、該当する単語群（１つ、または複数の単語）を順次抽出し、検索対象複合語候補格納バッファ部２０７に格納する。さらに、検索対象複合語候補バッファ部２０７に格納されている単語群から最大結合数バッファ部２０４に格納されている結合数以下の複合語を抽出し、検索対象抽出複合語格納バッファ部２０８に格納する。

【００２４】検索対象単語出現頻度算出部１０９は、検索対象単語複合語抽出部２０８により抽出された個々の単語や複合語について、検索対象文書中での出現頻度を算出し、これを検索対象文書の単語情報として検索対象単語情報格納バッファ部２０９に格納する。

【００２５】検索対象単語情報書き込み部１１０は、検索対象単語情報格納バッファ部２０９に格納されている検索対象文書の単語情報を外部記憶装置４に格納する。

【００２６】検索キー文書入力部１１１は、入力装置１から入力された検索キー文書の情報を検索キー文書格納バッファ部２１０に格納する。

【００２７】検索キー単語切り出し部１１２は、検索キー文書格納バッファ部２１０に格納されている検索キー文書からの単語切り出しを行う。そして、その検索キー文書から抽出される全ての単語とその品詞を検索キー文書全文分割単語格納バッファ部２１１に格納する。単語の切り出しは形態素解析などにより行、その文書から抽出される単語の品詞情報を「名詞」、「サ変名詞」、「形容詞」などで表現する。　検索キー単語複合語抽出部１１３は、検索キー全文分割単語格納バッファ部２１１に格納されている全ての単語とその品詞情報の中から、選択品詞情報バッファ部２０２に格納されている品詞情報を参照して、該当する単語群（１つ、または複数の単語）を順次抽出し、検索キー複合語候補格納バッファ部２１２に格納する。さらに、検索キー複合語候補格納バッファ部２１２に格納されている単語群から最大結合数バッファ部に格納されている結合数以下の複合語を抽出し、検索キー抽出複合語格納バッファ部２１３に格納する。

【００２８】検索キー単語出現頻度算出部１１４は、検索キー単語複合語抽出部１１３により抽出された個々の単語や複合語について、検索キー文書中での出現頻度を算出し、これを検索キー文書の単語情報として検索対象単語情報格納バッファ部２１４に格納する。

【００２９】検索対象単語情報読み出し部１１５は、外部記憶装置４に格納されている各検索対象文書の単語情報（単語の出現頻度情報）を１文書毎に呼び出し、検索対象単語情報格納バッファ部２０９に格納する。

【００３０】共通単語抽出部１１６は、検索キー単語情報格納バッファ部２１４に格納されている検索キー文書の単語情報と検索対象単語情報格納バッファ部２０９に

格納されている検索対象文書の単語情報とを比較して、一致する単語の種類と出現頻度情報を共通単語情報格納バッファ部２１５に格納する。

【００３１】類似度算出部１１７は、共通単語情報格納バッファ部２１５に格納されている情報に基づき、検索キー文書と検索対象文書との類似度を算出し、その類似度を算出類似度格納バッファ部２１６に格納する。

【００３２】検索結果出力部１１８は、算出類似度格納バッファ部２１６に格納されている各検索対象文書の類似度値を適宜並べ替えて、検索結果出力バッファ部２１７に格納し、さらに検索結果出力バッファ部２１７の内容を表示装置３に出力する。次に、本実施形態の類似文書検索装置の動作を説明する。

【００３３】最初に検索対象文書データベースの作成手順を図３のフローチャートにより説明する。

【００３４】まず、初期化部１０１により全メモリ部の初期化を行い、外部記憶装置４の品詞辞書と不要語辞書の情報をそれぞれ品詞情報バッファ部２０１、不要語情報バッファ部２０３に格納する（ステップ３０１）。品詞情報バッファ部２０１の構成を図９に、不要語情報バッファ部２０３の構成を図１１にそれぞれ示す。

【００３５】続いて抽出対象品詞設定部１０４が起動され、入力装置１を通じてユーザより抽出する単語の品詞選択の入力を受け付けて、図１０に示すように、抽出対象品詞を選択品詞情報バッファ部２０２に格納する。また、同様に最大結合数設定部１０５が起動され、入力装置１を通じてユーザより抽出する複合語の最大結合数値の入力を受け付けて、図１２に示すように、最大結合数バッファ部２０４に格納する（ステップ３０２）。

【００３６】これらの設定が完了すると、検索対象文書読み出し部１０６が外部記憶装置４から複数のテキスト文書を読み出し、検索対象文書格納バッファ２０５に検索対象文書として格納する（ステップ３０３）。具体例として、例えば、図１３に示すような内容のテキスト文書を検索対象文書として格納されたとする。

【００３７】次に、検索対象単語切り出し部１０７が、検索対象文書格納バッファ部２０５に格納されている検索対象文書について、形態素解析などによって単語の切り出しを行い、切り出した単語とその品詞情報を、図１４に示すように、検索対象全文分割単語格納バッファ部２０６に格納する（ステップ３０４）。

【００３８】続いて検索対象単語複合語抽出部１０８が起動される。検索対象単語複合語抽出部１０８は、検索対象全文分割単語格納バッファ部２０６に格納されている当該検索対象文書の全単語とその品詞情報、選択品詞情報バッファ２０２を参照し、該当する単語（１個以上）を、図１５に示すように、検索対象複合語候補格納バッファ部２０７に格納する。

【００３９】さらに、検索対象単語複合語抽出部１０８は、検索対象複合語候補格納バッファ部２０７に格納されている１個、または複数の単語群、不要語情報バッファ部２０３の不要語情報、そして最大結合数バッファ部２０４の最大結合数値を参照し、１以上かつ最大結合数以下の単語の結合からなる複合語を検索対象文書の中から抽出し、図１６に示すように、検索対象抽出複合語格納バッファ部２０８に全て格納する（ステップ３０５）。なお、ここで抽出される複合語には、単独の単語、つまり結合数１の単語も含むものとする。

【００４０】ここで、検索対象単語複合語抽出部１０８により、単語、複合語が少なくとも１個以上抽出された場合（ステップ３０６）、検索対象単語出現頻度算出部１０９が起動される。検索対象単語出現頻度算出部１０９は、検索対象抽出複合語格納バッファ部２０８に格納されている複合語について、当該検索対象文書中での出現頻度を複合語別に累積し、図１７に示すように、検索対象単語情報格納バッファ部２０９に順次格納する（ステップ３０７）。検索対象語情報格納バッファ部２０９において、複合語（単語も含む）と頻度とは対応して登録されており、例えば、単語「住所録」は当該文書中に４回出現していることを表す。

【００４１】以上の複合語抽出処理と抽出単語出現頻度算出処理は当該文書の文末まで行われる（ステップ３０８）。

【００４２】当該文書の複合語抽出処理と抽出単語出現頻度算出処理が終了すると、図１７に示す検索対象単語情報格納バッファ部２０９に格納された情報は、検索対象文書のデータベースとして外部記憶装置４に蓄積される（テップ３０９）。

【００４３】これで１検索対象文書のデータベースへの蓄積が終了するが、検索対象文書格納バッファ部２０５にまだ検索対象文書が残っている場合、ステップ３０４にもどって、前記同様の文書データベース生成が行われる。検索対象文書が残っていない場合、データベースの生成は終了する。

【００４４】ここで、検索対象単語複合語抽出部１０８による単語複合語抽出（ステップ３０５）の手順を図５、図６、図１４～図１６を使って詳しく説明する。

【００４５】まず、初期化として現結合数に０を代入する（ステップ５１０）。複合語候補となる結合単語を抽出するため、図１４に示す検索対象全文分割単語格納バッファ２０６に記憶されている単語に対応する品詞情報と、図１０に示す選択品詞情報バッファ部２０２にある品詞情報とを比較し（ステップ５０２）、検索対象全文分割単語格納バッファ２０６に記憶されている単語が対象品詞の単語であった場合、図１５に示すように、当該単語を検索対象複合語候補格納バッファ部２０７に格納し（ステップ５０３）、現結合数に１を加える（ステップ５０４）。対象品詞の単語をすべて抽出したらステップ５０５に移る。

【００４６】ステップ５０５では現結合数を調べ、現結

合数が０より大きかった場合は処理を続行し、０であった場合は複合語抽出処理を終了する。

【００４７】現結合数が０より大きい場合は、続いて、複合語抽出のための先頭カウンタと結合数カウンタにそれぞれ１をセットして初期化を行う（ステップ５０６）。

【００４８】ここから、複合語の抽出が、結合数カウンタが最大結合数バッファ部２０４が示す値になるまで以下のように行われる（ステップ５０７）。

【００４９】先頭カウンタが示す検索対象複合語候補格納バッファ部２０７の単語から結合数カウンタの示す単語数の複合語を抽出できる場合（ステップ５０８）、ステップ５０９からステップ５１２にかけて不要語チェックを行う。不要語チェックは、当該複合語を構成する単語と図１１に示す不要語情報バッファ部２０３の単語とを全て比較し、不要語に該当するものがあった場合、その不要語を含む複合語を抽出の対象としない処理を行う（ステップ５１３）。

【００５０】なお、ここでは複合語の一要素となる単語が不要語であった場合、複合語抽出の対象としない処理を行ったが、そうした不要語が複合語の頭に接頭する、あるいは、末尾に接尾する場合にだけ、複合語抽出の対象としない処理を行うようにしてもよい。

【００５１】不要語にあたる単語が、抽出された複合語に含まれない場合は、その複合語を検索対象抽出複合語格納バッファ部２０８に格納する（ステップ５１４）。

【００５２】このときの結合数カウンタが示す単語数の複合語を図１５に示す検索対象複合語候補格納バッファ２０７からすべて抽出する（ステップ５１４）。すべて抽出したら、結合数カウンタに１を加え（ステップ５１５）、先頭から新たな結合数カウンタが示す結合数の複合語の抽出を行う（ステップ５１６）。

【００５３】そして結合数カウンタが最大結合数バッファ部２０４が示す値を超えた場合、複合語抽出を終了する（ステップ５０７）。

【００５４】次に、類似文書の検索手順を、図４のフローチャートにより説明する。

【００５５】まず、初期化部１０１により全メモリ部を初期化し、外部記憶装置４の品詞辞書と不要語辞書の情報をそれぞれ品詞情報バッファ部２０１、不要語情報バッファ部２０３に格納する。（ステップ４０１）。続いて抽出対象品詞設定部１０４が起動され、入力装置１を通じてユーザより抽出する単語の品詞選択の入力を受け付けて抽出対象品詞を選択品詞情報バッファ部２０２に格納する。また、同様に最大結合数設定部１０５が起動され、入力装置１を通じてユーザより抽出する複合語の最大結合数値の入力を受け付けて最大結合数バッファ部２０４に格納する（ステップ４０２）。

【００５６】続いて、検索キー文書入力部１１１が起動され、入力装置１を通じてユーザより検索キーとなる文書の入力を受け付けて検索キー文書格納バッファ部２１０に格納する（ステップ４０３）。具体例として、例えば、図１８に示すような内容のテキスト文書を検索キー文書として格納したとする。

【００５７】次に、検索キー単語切り出し部１１２が、検索キー文書格納バッファ部２１０に格納されている検索キー文書について、形態素解析などによって単語の切り出しを行い、切り出した単語とその品詞情報を検索キー全文分割単語格納バッファ部２１１に格納する（ステップ４０４）。

【００５８】そして、検索キー単語複合語抽出部１１３が起動される。検索キー単語複合語抽出部１１３は、検索キー全文分割単語格納バッファ部２１１に格納されている当該検索キー文書の全単語とその品詞情報、選択品詞情報バッファ２０１，２０２を参照し、該当する単語（１個以上）を検索キー複合語候補格納バッファ部２１２に格納する。さらに、検索キー単語複合語抽出部１１３は、検索キー複合語候補格納バッファ部２１２に格納されている１個、または複数の単語群、不要語情報バッファ部２０３の不要語情報、そして最大結合数バッファ２０４の最大結合数値を参照し、１以上かつ最大結合数以下の単語が結合した複合語を検索キー文書の中から抽出し、これらを検索キー抽出複合語格納バッファ部２１３に全て格納する（ステップ４０５）。

【００５９】なお、ここで抽出される複合語には、単独の単語、つまり結合数１の単語も含むものとする。

【００６０】ここで、検索キー単語複合語抽出部１１３により、単語、複合語が少なくとも１個以上抽出された場合（ステップ４０６）、検索キー単語出現頻度算出部１１４が起動される。検索キー単語出現頻度算出部１１４は、検索キー抽出複合語格納バッファ部２１３に格納されている複合語について、当該検索キー文書中での出現頻度を複合語別に累積し、検索キー単語情報格納バッファ部２１４に順次格納する（ステップ４０７）。図２２に検索キー単語情報格納バッファ部２１４の格納例を示す。この検索キー単語情報格納バッファ部２１４において、複合語（単語も含む）と頻度は対応しており、例えば、単語「筆文字」は当該文書中に３回出現していることを表す。

【００６１】この複合語抽出処理と抽出単語出現頻度算出処理を当該文書の文末まで行う（ステップ４０８）。これで検索キー文書の単語情報の生成が終了する。

【００６２】次に、検索対象単語情報読み出し部１１５が、外部記憶装置４に格納されている各検索対象文書の単語情報を１文書毎に読み込み、検索対象単語情報格納バッファ２０９に格納する（ステップ４０９）。

【００６３】続いて、共通単語抽出部１１６が起動され、検索対象単語情報格納バッファ部２０６と検索キー単語情報格納バッファ部２１４とに共通して格納されている単語、複合語を共通単語情報格納バッファ部２１５

に格納する。具体例を図２３に示す。図１７の検索対象単語情報格納バッファ部２０９と図２２の検索キー単語情報格納バッファ部２１４に共通する単語（複合語）として、「宛名印刷」が抽出され、この「宛名印刷」とその頻度が検索キー側、検索対象側それぞれ１、２というように対応づけて格納する（ステップ４１０）。

【００６４】次に、類似度算出部１１７が、共通単語情報格納バッファ部２１５に格納されている頻度情報に基づき検索キーと検索対象文書との類似度を空間ベクトル法などにより算出し、その類似度値を算出類似度格納バッファ部２１６に格納する（ステップ４１１）。例えば、図２４に示すように、各検索対象文書ごとの類似度が算出類似度格納バッファ部２１６に格納される。

【００６５】全ての検索対象文書について類似度計算が終了すると（ステップ４１２）、検索結果出力部１１８は、算出類似度格納バッファ部２１６に格納されている各検索対象文書ごとの類似度を類似度が高い順に並べ替えて検索結果出力バッファ部２１７に格納し、そのバッファの内容を表示装置３に出力する。出力結果は、例えば、図２６が示すような形で出力される（ステップ４１３）。なお、図２６では類似度値に閾値を設けて表示しているが、類似度値に一定の閾値を設けて、検索結果として表示する検索対象文書の量を制限できるようにしてもよい。

【００６６】これで１検索キー文書の類似文書検索は終了するが、新たに検索キー文書がある場合、ステップ４０２に戻って、同様な処理を行う。検索キー文書がなければ検索処理はこれで終了する（ステップ４１４）。

【００６７】なお、この類似文書検索処理における単語複合語抽出の手順は、検索対象文書データベース作成における処理と対象となる文書による処理部、バッファ部の違いはあるが全く同様の処理である。

【００６８】以上のように、本実施形態の類似文書検索装置では、検索キー文書あるいは検索対象文書に、例えば「筆文字宛名印刷機能」といった複合語が存在するならば、最大結合数を３としたとき「筆文字宛名」「文字宛名印刷」「宛名印刷機能」「筆文字」「文字宛名」「宛名印刷」「印刷機能」などの各所において部分的に連続した複合語が抽出される。このように、特定の内容の文書を特徴付ける異なる複合語を漏れなく抽出することができるので、文書間のより妥当な類似度を計算でき、ユーザの意図する精度の高い類似文書検索を行うことができる。また、検索対象となる文書全体から抽出される単語種の総数が少なくなり、データベースの規模を縮小することができる。

【００６９】さらに、本実施形態の類似文書検索装置では、文書中から複合語を抽出するとき、ユーザにより指定された不要語の単語を含む複合語を無効なものとすることで、ユーザの意図する類似文書検索をさらに高精度に行うことが可能になる。

【００７０】なお、以上説明した類似文書検索装置は，例えば、汎用的なハードウェア環境に、フロッピーディスク、ＣＤ－ＲＯＭなどの記録媒体に記録されたアプリケーションプログラムを追加することによっても提供することが可能である。

【００７１】

【発明の効果】以上説明したように、本発明によれば、文書から抽出された単語の結合により構成される複合語のうち、指定された上限値以下の数の単語により構成されるすべての複合語を検索キー文書および検索対象文書から抽出することで、文書を特徴付ける複合語を漏れなく抽出することができ、検索キー文書と検索対象文書との類似度をより高精度に算出することができる。また、予め指定された不要語の単語を含む複合語を無効なものとすることで、ユーザの意図をさらに反映したより一層高精度な類似文書検索を行うことが可能になる。

【図面の簡単な説明】

【図１】本発明の実施形態である類似文書検索装置のハードウェア構成を示すブロック図

【図２】図１の制御装置の内部構成を示すブロック図

【図３】本実施形態の類似文書検索装置の検索対象文書データベース生成の動作手順を示すフローチャート

【図４】本実施形態の類似文書検索装置の類似文書検索の動作手順を示すフローチャート

【図５】複合語抽出処理の詳細な動作手順を示すフローチャート

【図６】図５と同じく複合語抽出処理の詳細な動作手順を示すフローチャート

【図７】品詞辞書の内容を示す図

【図８】不要語辞書の内容を示す図

【図９】品詞情報バッファの内容を示す図

【図１０】選択品詞情報バッファの内容を示す図

【図１１】不要語情報バッファの内容を示す図

【図１２】最大結合数バッファの内容を示す図

【図１３】検索対象文書格納バッファの内容を示す図

【図１４】検索対象全文分割単語格納バッファの内容を示す図

【図１５】検索対象複合語候補格納バッファの内容を示す図

【図１６】検索対象抽出複合語格納バッファの内容を示す図

【図１７】検索対象単語情報格納バッファの内容を示す図

【図１８】検索キー文書格納バッファの内容を示す図

【図１９】検索キー全文分割単語格納バッファの内容を示す図

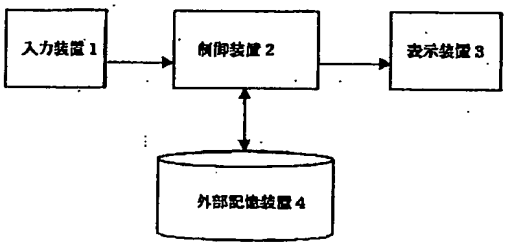
【図２０】検索キー複合語候補格納バッファの内容を示す図

【図２１】検索キー抽出複合語格納バッファの内容を示す図

【図２２】検索キー単語情報格納バッファの内容を示す図
【図２３】共通単語情報格納バッファの内容を示す図
【図２４】算出類似度格納バッファの内容を示す図
【図２５】検索結果出力バッファの内容を示す図
【図２６】類似文書検索結果の出力例を示す図
【符号の説明】

１００　制御部
１０１　初期化部
１０２　入力部
１０３　出力部
１０４　抽出対象品詞設定部
１０５　最大結合数設定部
１０６　検索対象文書読み出し部
１０７　検索対象単語切り出し部
１０８　検索対象単語複合語抽出部
１０９　検索対象単語出現頻度算出部
１１０　検索対象単語情報書き込み部
１１１　検索キー文書入力部
１１２　検索キー単語切り出し部
１１３　検索キー単語複合語抽出部
１１４　検索キー単語出現頻度算出部
１１５　検索対象単語情報読み出し部
１１６　共通単語抽出部
１１７　類似度算出部
１１８　検索結果出力部
２００　メモリ部
２０１　品詞情報バッファ部
２０２　選択品詞情報バッファ部
２０３　不要語情報バッファ部
２０４　最大結合数バッファ部
２０５　検索対象文書格納バッファ部
２０６　検索対象全文分割単語格納バッファ部
２０７　検索対象複合語候補格納バッファ部
２０８　検索対象抽出複合語格納バッファ部
２０９　検索対象単語情報格納バッファ部
２１０　検索キー文書格納バッファ部
２１１　検索キー全文分割単語格納バッファ部
２１２　検索キー複合語候補格納バッファ部
２１３　検索キー抽出複合語格納バッファ部
２１４　検索キー単語情報格納バッファ部
２１５　共通単語情報格納バッファ部
２１６　算出類似度格納バッファ部
２１７　検索結果出力バッファ部

【図１】



【図１１】

不要語情報バッファ

下記↓
後述↓
詳述↓
上記↓
上述↓
:

(↓：文字終端)

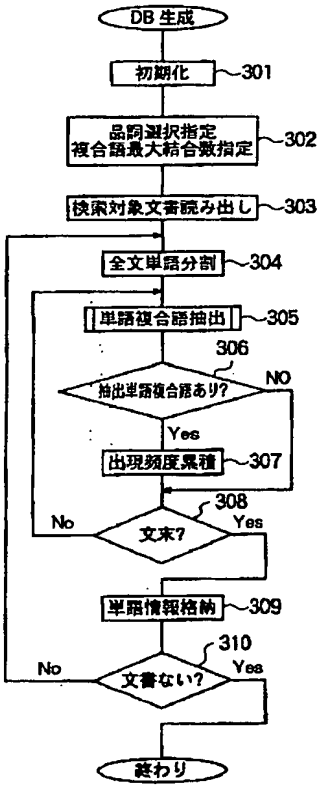
【図１３】

検索対象文書格納バッファ

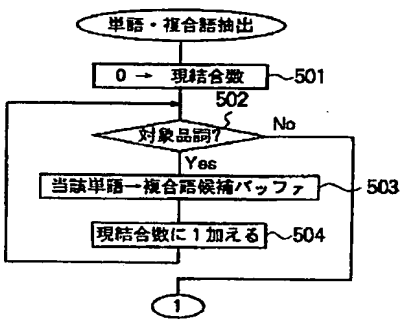
筆文字宛名印刷機能を中心とするソフトで、住所録データベースのデータを変換するためのツールが……↓

(↓：文字終端)

【図３】



【図５】



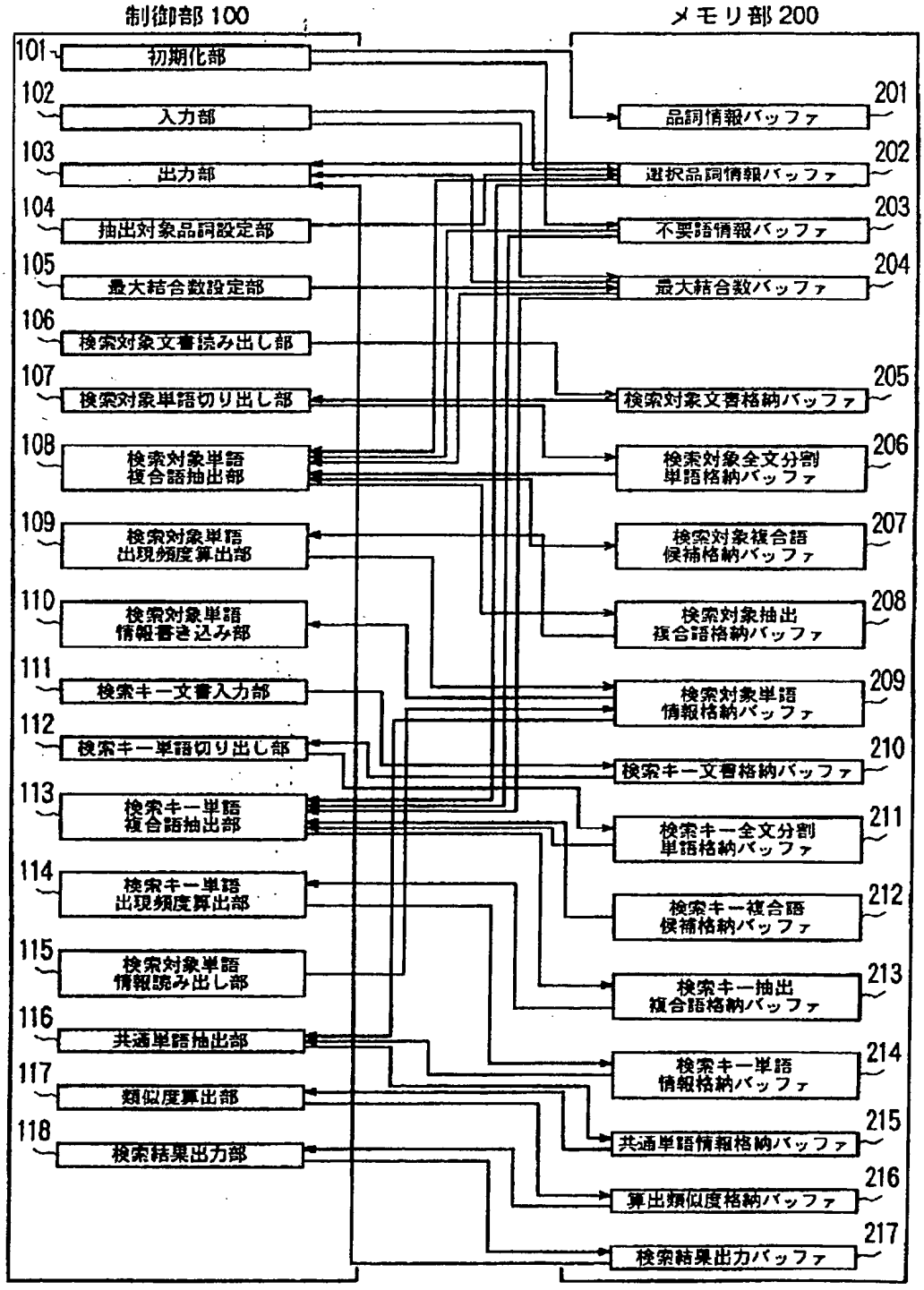
【図１９】

検索キー全文分割単語格納バッファ

筆↓	名詞↓
文字↓	名詞↓
に↓	格助詞↓
:	:
上記↓	名詞↓
宛名↓	名詞↓
印刷↓	サ変名詞↓
や↓	格助詞↓
デジカメ↓	名詞↓
:	:

(↓：文字終端)

【図２】



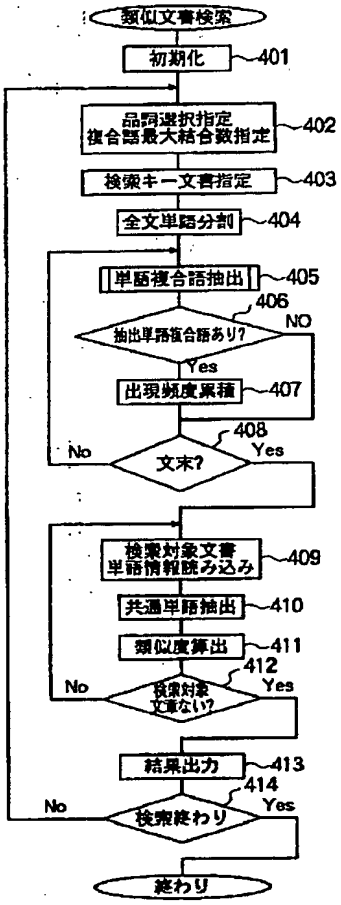
【図１２】

最大結合数バッファ

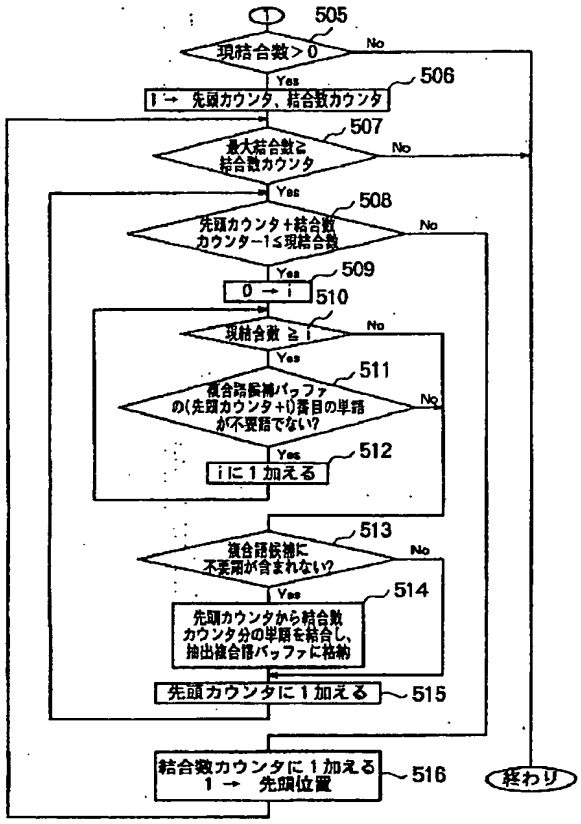
3↓

（↓：文字終端）

【図４】



【図６】



【図７】

【図８】

品詞辞書

品詞
名詞↓
サ変名詞↓
動詞↓
形容詞↓
形容動詞↓
・・

（↓：文字終端）

不要語辞書

登録単語
下記↓
後述↓
詳述↓
上記↓
上述↓
図中↓
前記↓
：：

（↓：文字終端）

【図１０】

【図１５】

選択品詞情報バッファ

名詞↓
サ変名詞↓
形容詞↓
形容動詞↓

（↓：文字終端）

検索対象複合語候補格納バッファ

筆↓	名詞↓
文字↓	名詞↓
宛名↓	名詞↓
印刷↓	サ変名詞↓
機能↓	名詞↓

（↓：文字終端）

【図９】

品詞情報バッファ

名詞↓
サ変名詞↓
形容詞↓
形容動詞↓
:
副詞↓
格助詞↓

(↓：文字終端)

【図１４】

検索対象全文分割単語格納バッファ

筆↓	名詞↓
文字↓	名詞↓
宛名↓	名詞↓
印刷↓	サ変名詞↓
機能↓	名詞↓
も↓	格助詞↓
中心↓	名詞↓
と↓	格助詞↓
:	:

(↓：文字終端)

【図１６】

検索対象抽出複合語格納バッファ

筆文字宛名↓
文字宛名印刷↓
:
宛名印刷機能↓
:
宛名印刷↓
:

(↓：文字終端)

【図１７】

検索対象単語情報格納バッファ

単語	頻度
筆文字宛名↓	2↓
宛名印刷↓	2↓
:	:
住所録↓	4↓
:	:
変換↓	8↓
:	:

(↓：文字終端)

【図１８】

【図２０】

【図２１】

検索キー文書格納バッファ

筆文字フォントによる上記宛名印刷やデジカメ画像からの映像取り込み、住所録のデータベースのインポート／エクスポート機能が充実.....↓

(↓：文字終端)

検索キー複合語候補格納バッファ

上記↓	名詞↓
宛名↓	名詞↓
印刷↓	サ変名詞↓

(↓：文字終端)

検索キー抽出複合語格納バッファ

宛名印刷↓
宛名↓
印刷↓

(↓：文字終端)

【図２２】

【図２３】

【図２５】

検索キー単語情報格納バッファ

単語	頻度
筆文字↓	3↓
宛名印刷↓	1↓
:	:
住所録↓	6↓
:	:
機能↓	4↓

共通単語情報格納バッファ

単語	検索キー側頻度	検索対象側頻度
宛名印刷↓	1↓	2↓
住所録↓	6↓	4↓
変換↓	8↓	3↓
機能↓	4↓	1↓
:	:	:

(↓：文字終端)

検索結果出力バッファ

類似検索対象文書
3↓
51↓
289↓
:

(↓：文字終端)

【図２４】

【図２６】

算出類似度格納バッファ

検索対象文書番号	類似度
1↓	0.03198↓
2↓	0.16771↓
3↓	0.49534↓
:	:

(↓：文字終端)

検索結果出力例

類似文書検索結果

<文書番号>
3
51
289
:

フロントページの続き

(72)発明者　仁科　卓哉
東京都青梅市新町1381番地１　東芝コンピュータエンジニアリング株式会社内

(72)発明者　中本　幸夫
東京都青梅市新町1381番地１　東芝コンピュータエンジニアリング株式会社内

Information Retrieval

William B. Frakes
Ricardo Baeza-Yates
Editors

Data Structures & Algorithms

PTR

ISBN 0-13-463837-9
90000
9 780134 638379

Information Retrieval

Data Structures & Algorithms

Edited by

William B. Frakes
Software Engineering Guild

Ricardo Baeza-Yates
University of Chile

Prentice Hall PTR, Upper Saddle River, New Jersey 07458

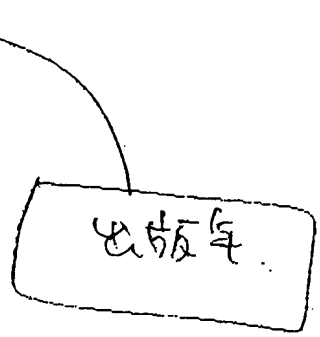
Library of Congress Cataloging-in-Publication Data

Information retrieval : data structures and algorithms / edited by William B. Frakes, Ricardo Baeza-Yates.
p. cm.
Includes bibliographical references and index.
ISBN 0-13-463837-9
1. Data structures (Computer science). 2. Computer algorithms.
I. Frakes, William B. (William Bruce), 1952– II. Baeza-Yates, R. (Ricardo)
QA76.9.D351543 1992
005—dc20 92-8197
CIP

Editorial production and interior design: *bookworks*
Acquisitions editor: *Greg Doench*
Managing editor: *Sophie Papanikolaou*
Cover designer: *Bruce Kenselaar*
Copy editor: *Karen Verde*
Editor-in-Chief: *Bernard Goodwin*
Prepress buyer: *Mary McCartney*
Manufacturing buyer: *Susan Brunke*
Indexer: *Word Finders*

Cover art credited to: Piet Mondrian. Composition in Brown and Gray. (1913–14). Oil on canvas. $33\frac{3}{4} \times 29\frac{3}{4}$". Collection, the Museum of Modern Art, New York. Purchase. Photograph © 1993 The Museum of Modern Art, New York.

Prentice-Hall, Inc.
A Simon & Schuster Company
Upper Saddle River, New Jersey 07458



Printed in the United States of America

10 9 8 7 6 5

ISBN 0-13-463837-9

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada, Inc., *Toronto*
Prentice-Hall Hispanoamericana S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

Cont

FOREW

PREFAC

INTROD

FILE STR

```
    for ( i = 0; i NO_STACKS; i++ ) {  /* Allocate stack space. */
      if ( stacks[i].stackSize = 0 ) stacks[i].stackSize = DEF_SIZE;
        stacks[i].stackRep =
          (vertexType**) owncalloc( stacks[i].stackSize, sizeof(vertexType*)
);
    }

    heap.heapSize = no_vertices - sum - (int) 2 * no_vertices * Pr0;
    if ( heap.heapSize = 0 ) heap.heapSize = DEF_SIZE;
    heap.heapRep =              /* Allocate heap space. */
        (heapCell*) owncalloc( heap.heapSize, sizeof(heapCell) );
}
```


Information Retrieval, 1992

14

Ranking Algorithms

Donna Harman
National Institute of Standards and Technology

Abstract

This chapter presents both a summary of past research done in the development of ranking algorithms and detailed instructions on implementing a ranking type of retrieval system. This type of retrieval system takes as input a natural language query without Boolean syntax and produces a list of records that "answer" the query, with the records ranked in order of likely relevance. Ranking retrieval systems are particularly appropriate for end-users.

14.1 INTRODUCTION

Boolean systems were first developed and marketed over 30 years ago at a time when computing power was minimal compared with today. Because of this, these systems require the user to provide sufficient syntactical restrictions in their query to limit the number of documents retrieved, and those retrieved documents are not ranked in order of any relationship to the user's query. Although the Boolean systems offer very powerful on-line search capabilities to librarians and other trained intermediaries, they tend to provide very poor service to end-users, particularly those who use the system on an infrequent basis (Cleverdon 1983). These end-users are likely to be familiar with the terminology of the data set they are searching, but lack the training and practice necessary to get consistently good results from a Boolean system because of the complex query syntax required by these systems. The ranking approach to retrieval seems to be more oriented toward these end-users. This approach allows the user to input a simple query such as a sentence or a phrase (no Boolean connectors) and retrieve a list of documents ranked in order of likely relevance.

The main reason the natural language/ranking approach is more effective for end-users is that all the terms in the query are used for retrieval, with the results being ranked based on co-occurrence of query terms, as modified by statistical term-weighting (to be explained later in the chapter). This method eliminates the often-wrong Boolean syntax used by end-users, and provides some results even if a query term is incorrect, that is, it is not the term used in the data, it is misspelled, and so on. The ranking methodology also works well for the complex queries that may be

difficult for end-users to express in Boolean logic. For example, "human factors and/or system performance in medical databases" is difficult for end-users to express in Boolean logic because it contains many high- or medium-frequency words without any clear necessary Boolean syntax. The ranking method would do well with this query.

This chapter describes the implementation of a ranking system and is organized in the following manner. Section 14.2 shows a conceptual illustration of how ranking is done. Section 14.3 presents various theoretical models used in ranking and reviews past experiments using these models. Section 14.4 describes results from several experiments directly comparing various ranking schemes, along with brief discussions of the type of ranking methods found in the few operational systems that have used this retrieval technique. Section 14.5 summarizes the results from sections 14.3 and 14.4, presenting a series of recommended ranking schemes for various situations. Section 14.6 describes the implementation of a basic ranking retrieval system, with section 14.7 showing possible variations to this scheme based on retrieval environments. Section 14.8 discusses some topics closely related to ranking and provides some suggestions for further reading in these areas, and section 14.9 summarizes the chapter.

The terms "record" and "document" are used interchangeably throughout this chapter, as are the terms "data set," "database," and "collection," depending on the retrieval environment being discussed.

14.2 HOW RANKING IS DONE

Assume that a given textual data set uses n unique terms. A document can then be represented by a vector $(t_1, t_2, t_3, \ldots, t_n)$, where t_i has a value of 1 if term i is present, and 0 if term i is absent in the document. A query can be represented in the same manner. Figure 14.1 shows this representation for a data set with seven unique terms.

The top section of Figure 14.1 shows the seven terms in this data set. The second section shows a natural language query and its translation into a conceptual vector, with 1's in vector positions of words included in the query, and 0's to indicate a lack of those words. For example, the first "1" indicates the presence of the word "factor," the second "1" indicates the presence of the word "information," the first "0" indicates the absence of the word "help." The third section of Figure 14.1 shows a similar conceptual representation of three documents in this data set. To determine which document best matches the query, a simple dot product of the query vector and each document vector is made (left side of the fourth section) and the results used to rank the documents.

It is possible to perform the same operation using weighted vectors as shown in the right side of the bottom section of Figure 14.1. In the example, each term is weighted by the total number of times it appears in the record, for example, "factors" appears twice in document 1, "information" appears three times in document 1, and so on. These term-weights could reflect different measures, such as the scarcity of a term in the data set (i.e., "human" probably occurs less frequently than "systems" in a computer science data set), the frequency of a term in the given document (as shown in the example), or some user-specified term-weight. This term-weighting usually provides substantial improvement in the ranking.

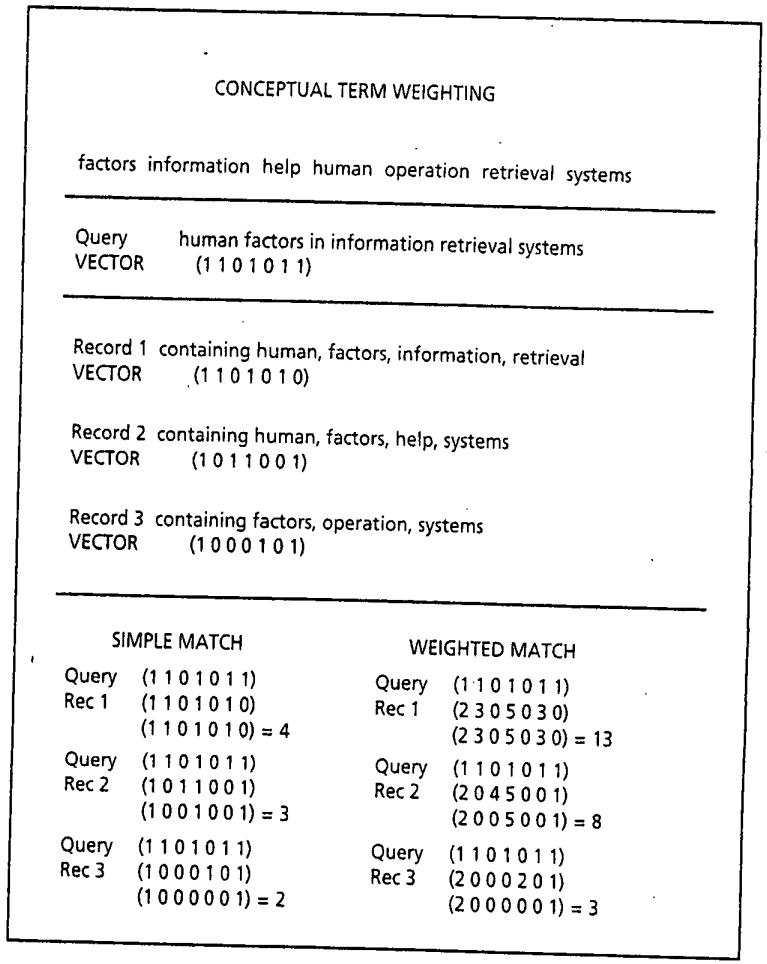


Figure 14.1 A simple illustration of statistical ranking

14.3 RANKING MODELS AND EXPERIMENTS WITH THESE MODELS

In 1957 Luhn published a paper proposing a statistical approach to searching literary information. He suggested that "the more two representations agreed in given elements and their distribution, the higher would be the probability of their representing similar information." Maron and Kuhns (1960) went much further by suggesting how to actually weight terms, including some small-scale experiments in term-weighting. The information retrieval research community has continued to develop many models for the ranking technique over the last 30 years (for an overview, see Belkin and Croft [1987]). Although it is not necessary to understand the theoretical models involved in ranking in detail in order to implement a ranking retrieval system, it is helpful to know about them as they have provided a framework for the vast majority of retrieval experiments that contributed to the development of the ranking techniques used today.

All the experimental results presented in the models are based on using standard test collections and using standard recall and precision measures for evaluation. Results are presented in a roughly chronological order to provide some sense of the development of knowledge about ranking through these experiments.

Ranking models can be divided into two types: those that rank the query against individual documents and those that rank the query against entire sets of related documents. The first type of ranking model, those that rank individual documents against the query, covers several theoretical models, but the principle ones are the vector space model and the probabilistic model.

14.3.1 The Vector Space Model

The sample document and query vectors described in section 14.2 can be envisioned as an n-dimensional vector space, where n corresponds to the number of unique terms in the data set. A vector matching operation, based on the cosine correlation used to measure the cosine of the angle between vectors, can then be used to compute the similarity between a document and a query, and documents can be ranked based on that similarity.

$$similarity\ (d_j, q_k) = \frac{\sum_{i=1}^{n} (td_{ij} \times tq_{ik})}{\sqrt{\sum_{i=1}^{n} td_{ij}^2 \times \sum_{i=1}^{n} tq_{ik}^2}}$$

where

td_{ij} = the i^{th} term in the vector for document j
tq_{ik} = the i^{th} term in the vector for query k
n = the number of unique terms in the data set

This model has been used as the basis for many ranking retrieval experiments, in particular the SMART system experiments under Salton and his associates (1968, 1971, 1973, 1981, 1983, 1988). These ranking experiments started in 1964 at Harvard University, moved to Cornell University in 1968, and form a large part of the research in information retrieval. The SMART experiments cover many areas of information retrieval such as relevance feedback, clustering, and experiments with suffixing, synonyms, and phrases. Only those experiments dealing directly with term-weighting and ranking will be discussed here.

Early experiments (Salton and Lesk 1968; Salton 1971, p. 143) using the SMART system tested an overlap similarity function against the cosine correlation measure and tried simple term-weighting using the frequency of terms within the documents. These experiments showed that within-document frequency weighting improved performance over no term-weighting (in varying amounts depending on the test collection used). Further, they showed that use of the cosine correlation with frequency term-weighting provided better performance than the overlap similarity because of the automatic inclusion of document length normalization by the cosine similarity function (again, the results varied somewhat depending on the test collection used).

A second major set of experiments was done by Salton and Yang (1973) to further develop the term-weighting schemes. Of particular interest in these experiments were the term-weighting schemes relying on term importance within an entire collection rather than only within a given document. Clearly more weight should be given to query terms matching document terms that are rare within a collection. Salton and Yang were able to show significant performance improvement using a term-weighting scheme that combined the within-document frequency weighting with a new term-weighting scheme, the inverted document frequency (IDF; Sparck Jones 1972) that is based on the Zipf distribution of a term within the entire collection (see page 373 for the definition of IDF).

A recent paper by Salton and Buckley (1988) summarizes 20 years of SMART experiments in automatic term-weighting by trying 287 distinct combinations of term-weighting assignments, with or without cosine normalization, on six standard collections. Besides confirming that the best document term-weighting is provided by a product of the within-document term frequency and the IDF, normalized by the cosine measure, they show performance improvements using enhanced query term-weighting measures for queries with term frequencies greater than one. These weighting schemes are further discussed in section 14.5.

14.3.2 Probabilistic Models

Although a model of probabilistic indexing was proposed and tested by Maron and Kuhns (1960), the major probabilistic model in use today was developed by Robertson and Sparck Jones (1976). This model is based on the premise that terms that appear in previously retrieved relevant documents for a given query should be

given a higher weight than if they had not appeared in those relevant documents. In particular, they presented the following table showing the distribution of term t in relevant and nonrelevant documents for query q.

Document Indexing \ Document Relevance	+	−	
+	r	$n - r$	n
−	$R - r$	$N - n - R + r$	$N - n$
	R	$N - R$	N

N = the number of documents in the collection

R = the number of relevant documents for query q

n = the number of documents having term t

r = the number of relevant documents having term t

$$w^1 = \log \frac{\left(\frac{r}{R}\right)}{\left(\frac{n}{N}\right)} \qquad w^2 = \log \frac{\left(\frac{r}{R}\right)}{\left(\frac{n-r}{N-R}\right)}$$

$$w^3 = \log \frac{\left(\frac{r}{R-r}\right)}{\left(\frac{n}{N-n}\right)} \qquad w^4 = \log \frac{\left(\frac{r}{R-r}\right)}{\left(\frac{n-r}{N-n-R+r}\right)}$$

They then use this table to derive four formulas that reflect the relative distribution of terms in the relevant and nonrelevant documents, and propose that these formulas be used for term-weighting (the logs are related to actual use of the formulas in term-weighting).

Robertson and Sparck Jones also formally derive these formulas, and show that theoretical preference is for F4.

Formula F1 had been used by Barkla (1969) for relevance feedback in a SDI service and by Miller (1971) in devising a probabilistic search strategy for Medlars. It was also used by Sparck Jones (1975) in devising optimal performance yardsticks for test collections. Formula F4 (minus the log) is the term precision weighting measure proposed by Yu and Salton (1976).

Robertson and Sparck Jones used these four formulas in a series of experiments with the manually indexed Cranfield collection. They did experiments using all the relevance judgments to weight the terms to see what the optimal performance would be, and also used relevance judgments from half the collection to weight the terms for retrieval from the second half of the collection. In both cases, formula F4 was superior (closely followed by F3), with a large drop in performance between the optimal performance and the "predictive" performance, as would be expected. The experimental verification of the theoretical superiority of F4 provided additional weight to the importance of this new model.

The use of this theory as a predictive device was further investigated by Sparck Jones (1979a) who used a slightly modified version of F1 and F4 and again got much better results for F4 than for F1, even on a large test collection. Sparck Jones (1979b) tried using this measure (F4 only) in a manner that would mimic a typical on-line session using relevance feedback and found that adding the relevance weighting from only the first couple of relevant documents retrieved by a ranking system still produced performance improvements.

Work up to this point using probabilistic indexing required the use of at least a few relevant documents, making this model more closely related to relevance feedback than to term-weighting schemes of other models. In 1979 Croft and Harper published a paper detailing a series of experiments using probabilistic indexing without any relevance information. Starting with a probabilistic restatement of F4, they assume that all query terms have equal probability of occurring in relevant documents and derive a term-weighting formula that combines a weight based on the number of matching terms and on a term-weighting similar to the IDF measure.

$$similarity_{jk} = \sum_{i=1}^{Q} \left(C + \log \frac{(N - n_i)}{n_i}\right)$$

where

Q = the number of matching terms between document j and query k

C = a constant for tuning the similarity function

n_i = the number of documents having term i in the data set

N = the number of documents in the data set

Experimental results showed that this term-weighting produced somewhat better results than the use of the IDF measure alone. Being able to provide different values to C allows this weighting measure to be tailored to various collections. Setting C to 1 ranks the documents by IDF weighting within number of matches, a method that was suitable for the manually indexed Cranfield collection used in this study (because it can be assumed that each matching query term was very significant). C was set much lower in tests with the UKCIS2 collection (Harper 1980) because the terms were assumed to be less accurate, and the documents were very short (consisting of titles only).

Croft (1983) expanded his combination weighting scheme to incorporate within-document frequency weights, again using a tuning factor K on these weights to allow tailoring to particular collections. The results show significant improvement over both the IDF weighting alone and the combination weighting, with the scaling factor K playing a large part in tuning the weighting to different collections.

$$similarity_{jk} = \sum_{i=1}^{Q} (C + IDF_i) * f_{ij}$$

where

Q = the number of matching terms between document j and query k

IDF_i = the IDF weight for term i in the entire collection (see page 373 for the definition of IDF)

$$f_{ij} = K + (1 - K)\frac{freq_{ij}}{maxfreq_j}$$

where

$freq_{ij}$ = the frequency of term i in document j

K = a constant for adjusting the relative importance of the two weighting schemes

$maxfreq_j$ = the maximum frequency of any term in document j

The best value for K proved to be 0.3 for the automatically indexed Cranfield collection, and 0.5 for the NPL collection, confirming that within-document term frequency plays a much smaller role in the NPL collection with its short documents having few repeating terms.

14.3.3 Other Models for Ranking Individual Documents

Several other models have been used in developing term-weighting measures. The inverted document frequency measure heavily used in implementing both the vector space model and the probabilistic model was derived by Sparck Jones (1972) from observing the Zipf distribution curve for collection vocabulary. Others have tried more complex term distributions, most notably the 2-Poisson model proposed by Bookstein and Swanson (1974) and implemented and tested by Harter (1975) and Raghavan et al. (1983). This distribution model proved much less successful because of the difficulty in estimating the many parameters needed for implementation.

14.3.4 Set-Oriented Ranking Models

The most well known of the set-oriented models are the clustering models where a query is ranked against a hierarchically grouped set of related documents. This model is the subject of Chapter 16 and will not be further discussed here.

Models based on fuzzy set theory have been proposed (for a summary, see Bookstein [1985]) but have not received enough experimental implementations to be used in practice (except when combined with Boolean queries such as in the P-Norm discussed in Chapter 15).

The theory of rough sets has been applied to information retrieval (Srinivasan 1989) but similarly has not been developed far enough to be used in practice.

14.4 OTHER EXPERIMENTS INVOLVING RANKING

14.4.1 Direct Comparison of Similarity Measures and Term-Weighting Schemes

There have been several studies examining the various factors involved in ranking that have not been based on any particular model but have instead used some method of comparing directly various similarity measures and term-weighting schemes.

Sparck Jones (1973) explored different types of term frequency weightings involving term frequency within a document, term frequency within a collection, term postings within a document (a binary measure), and term postings within a collection, along with normalizing these measures for document length. She used four collections, with indexing generally taken from manually extracted keywords instead of using full-text indexing, and with all queries based on manual keywords. Her results showed that using the term frequency (or postings) within a collection always improved performance, but that using term frequency (or postings) within a document improved performance only for some collections. The various term-weighting schemes were not combined in this experiment.

McGill et al. (1979) examined the literature from different fields to select 67 similarity measures and 39 term-weighting schemes. He used these to rank results from Boolean retrievals using both controlled (manually indexed) and uncontrolled (full-text) indexing. For both controlled and uncontrolled vocabulary he found a significant difference in the performance of similarity measures, with a group of about 15 different similarity measures all performing significantly better than the rest. This group included both the cosine correlation and the inner product function used in the probabilistic models. The term-weighting results were more mixed, with

no significant difference found when using controlled vocabulary (i.e., term-weighting made no difference) and an overall significant difference found for uncontrolled vocabulary. There was a lack of significant difference between pairs of term-weighting measures for uncontrolled vocabulary, however, which could indicate that the difference between linear combinations of term-weighting schemes is significant but that individual pairs of term-weighting schemes are not significantly different.

A different approach was taken by Harman (1986). She selected four term-weighting factors proven important in past research and tried different combinations in order to arrive at an "optimum" term-weighting scheme. Full-text indexing was used on various standard test collections, with full-text indexing also done on the queries. The four factors investigated were: the number of matches between a document and a query, the distribution of a term within a document collection, the frequency of a term within a document, and the length of the document. Two different measures for the distribution of a term within a document collection were used, the IDF measure by Sparck Jones and a revised implementation of the "noise" measure (Dennis 1964; Salton and McGill 1983). Note that the use of noise here refers to how much a term can be considered useful for retrieval versus being simply a "noisy" term, and examines the concentration of terms within documents rather than just the number of postings or occurrences. She found that when using the single measures alone, the distribution of the term within the collection improved performance almost twice as much for the Cranfield collection as using only within-document frequency. Combining the within-document frequency with either the IDF or noise measure, and normalizing for document length improved results more than twice as much as using the IDF or noise alone in the Cranfield collection. Other collections showed less improvement, but the same relative merit of the term-weighting schemes was found. The noise measure consistently slightly outperformed the IDF (however with no significant difference).

14.4.2 Ranking Based on Document Structure

Some ranking experiments have relied more on document or intradocument structure than on the term-weighting described earlier. Bernstein and Williamson (1984) built a ranking retrieval system for a highly structured knowledge base, the Hepatitis Knowledge Base. Their ranking algorithms used not only weights based on term importance both within an entire collection and within a given document, but also on the structural position of the term, such as within summary paragraphs versus within text paragraphs. A very elaborate weighting scheme was devised for this experiment, tailored to the particular structure of the knowledge base. In SIBRIS, an operational information retrieval system (Wade et al. 1989), document and query structures are also used to influence the ranking, increasing term-weights for terms in titles of documents and decreasing term weights for terms added to a query from a thesaurus.

A very different approach based on complex intradocument structure was used in the experiments involving latent semantic indexing (Lochbaum and Streeter 1989). The indexing and retrieval were based on the singular value decomposition (related to factor analysis) of a term-document matrix from the entire document collection. This was combined with weighting using both a function of term frequency within a document (the root mean square normalization), and a function of term frequency within the entire collection (the noise or entropy measure, or alternatively the IDF measure). The results were generally superior to those using term-weighting alone, although further development is necessary before this approach is fast enough for use in large retrieval systems.

14.4.3 Ranking Techniques Used in Operational Systems

Several operational retrieval systems have implemented ranking algorithms as central to their search mechanism. The SIRE system, as implemented at Syracuse University (Noreault et al. 1977) built a hybrid system using Boolean searching and a vector-model-based ranking scheme, weighting by the use of raw term frequency within documents (for more on the hybrid aspects of this system, see section 14.7.3). A commercial outgrowth of this system, marketed as Personal Librarian, uses ranking based on different factors, including the IDF and the frequency of a term within a document. This system assigns higher ranks to documents matching greater numbers of query terms than would normally be done in the ranking schemes discussed experimentally.

The CITE system, designed as an interface to MEDLINE (Doszkocs 1982), ranked documents based solely on the IDF weighting, as no within-document frequencies were available from the MEDLINE files. For details on the search system associated with CITE, see section 14.7.2. The OPAKI project (Walker and Jones 1987) worked with on-line catalogs and also used the IDF measure alone. Although other small-scale operational systems using ranking exist, often their ranking algorithms are not clear from publications, and so these are not listed here.

14.5 A GUIDE TO SELECTING RANKING TECHNIQUES

In looking at results from all the experiments, some trends clearly emerge.

1. The use of term-weighting based on the distribution of a term within a collection always improves performance (or at minimum does not hurt performance). The IDF measure has been commonly used, either in its form as originally used, or in a form somewhat normalized.

$$IDF_i = \log_2 \frac{N}{n_i} + 1 \qquad \textit{(Sparck Jones 1972)}$$

$$IDF_i = \log_2 \frac{maxn}{n_i} + 1 \qquad \textit{(Sparck Jones 1979)}$$

$$IDF_i = \log_2 \frac{N - n_i}{n_i} \qquad \textit{(Croft and Harper 1979)}$$

where

N = the number of documents in the collection
n_i = the total number of occurrences of term i in the collection
$maxn$ = the maximum frequency of any term in the collection

A possible alternative is the noise or entropy measure tried in several experiments.

$$normalized\ noise_i = maxnoise - noise_i,$$

$$noise_i = \sum_{k=1}^{N} \frac{Freq_{ik}}{TFreq_i} \log_2 \frac{TFreq_i}{Freq_{ik}} \qquad \textit{(Harman 1986)}$$

$$entropy_i = 1 - \frac{\sum_{k=1}^{N} \frac{Freq_{ik}}{TFreq_i} \log_2 \frac{TFreq_i}{Freq_{ik}}}{\log_2 N} \qquad \textit{(Lochbaum \& Streeter 1989)}$$

where

N = the number of documents in the collection
$maxnoise$ = the highest noise of any term in the collection
$Freq_{ik}$ = the frequency of term i in document k
$TFreq_i$ = the total frequency of term i in the collection

2. The combination of the within-document frequency with the IDF weight often provides even more improvement. There are several reasons why this improvement is inconsistent across collections. First, it is very important to normalize the within-document frequency in some manner, both to moderate the effect of high-frequency terms in a document (i.e., a term appearing 20 times is not 20 times as important as one appearing only once) and to compensate for document length. This normalization has taken various forms in different experiments, but the lack of proper normalization techniques in some experiments has likely hidden possible improvements. A second reason for the inconsistent improvements found for within-document frequencies is the fact that some collections have very short documents (such as titles only) and therefore within-document frequencies play no role in these collections. Finally, the effects of within-document frequency may need to be tailored to collections, such as was done by Croft (1983) in using a sliding importance factor K, and by Salton and Buckley (1988) in providing different combination schemes for term-weighting. This tailoring seems to be particularly critical for manually indexed or controlled vocabulary data where use of within-document frequencies may even hurt performance. Either of the following normalized within-document frequency measures can be safely used.

$$cfreq_{ij} = K + (1 - K) \frac{freq_{ij}}{maxfreq_j} \qquad \textit{(Croft 1983)}$$

$$nfreq_{ij} = \frac{\log_2 (freq_{ij} + 1)}{\log_2 length_j} \qquad \textit{(Harman 1986)}$$

where

$freq_{ij}$ = the frequency of term i in document j
$maxfreq_j$ = the maximum frequency of any term in document j
$length_j$ = the number of unique terms in document j

3. Assuming within-document term frequencies are to be used, several methods can be used for combining these with the IDF measure. The combination recommended for most situations by Salton and Buckley is given below (a complete set of weighting schemes is presented in their 1988 paper).

$$similarity\ (Q,D) = \frac{\sum_{i=1}^{t} (w_{iq} \times w_{ij})}{\sqrt{\sum_{i=1}^{t} (w_{iq})^2 \times \sum_{i=1}^{t} (w_{ij})^2}}$$

where

$$w_{iq} = \left(0.5 + \frac{0.5\, freq_{iq}}{maxfreq_q}\right) \times IDF_i$$

and

$w_{ij} = freq_{ij} \times IDF_i$
$freq_{iq}$ = the frequency of term i in query q
$maxfreq_q$ = the maximum frequency of any term in query q
IDF_i = the IDF of term i in the entire collection
$freq_{ij}$ = the frequency of term i in document j

Salton and Buckley suggest reducing the query weighting w_{iq} to only the within-document frequency ($freq_{iq}$) for long queries containing multiple occurrences of terms, and to use only binary weighting of documents ($w_{ij} = 1$ or 0) for collections with short documents or collections using controlled vocabulary.

Many combinations of term-weighting can be done using the inner product. Whereas there is more flexibility available here than in the cosine measure, the need for providing normalization of within-document frequencies is more critical. Two possible combinations are given below that calculate the matching strength of a query to document j, with symbol definitions the same as those previously given.

$$similarity_j = \sum_{i=1}^{Q} ((C + IDF_i) \times cfreq_{ij}) \qquad (Croft\ 1983)$$

where

$$cfreq_{ij} = K + (1 - K)\frac{freq_{ij}}{maxfreq_j}$$

C should be set to low values (near 0) for automatically indexed collections, and to higher values such as 1 for manually indexed collections. K should be set to low values (0.3 was used by Croft) for collections with long (35 or more terms) documents, and to higher values (0.5 or higher) for collections with short documents, reducing the role of within-document frequency.

One alternative ranking using the inner product (but without adjustable constants) is given below.

$$similarity_j = \sum_{i=1}^{Q} \frac{\log_2 (freq_{ij} + 1) \times IDF_i}{\log_2 length_j} \qquad (Harman\ 1986)$$

4. It can be very useful to add additional weight for document structure, such as higher weightings for terms appearing in the title or abstract versus those appearing only in the text. This additional weighting needs to be considered with respect to the particular data set being used for searching. User weighting can also be considered as additional weighting, although this type of weighting has generally proven unsatisfactory in the past.
5. The use of relevance weighting after some initial retrieval is very effective. Relevance weighting is discussed further in Chapter 11 on relevance feedback.

14.6 DATA STRUCTURES AND ALGORITHMS FOR RANKING

This section will describe a simple but complete implementation of the ranking part of a retrieval system. Modifications of this implementation that enhance its efficiency or are necessary for other retrieval environments are given in section 14.7, with cross-references made to these enhancements throughout this section.

The implementation will be described as two interlocking pieces: the indexing of the text and the using (searching) of that index to return a ranked list of record identification numbers (ids). The index shown is a straightforward inverted file, created once per major update (thus only once for a static data set), and is used to provide the necessary speed for searching. Although it is possible to build a ranking retrieval system without some type of index (either by storing and searching all terms in a document or by using signature files in ranking such as described in section 14.8.5), the use of these indices improves efficiency by several orders of magnitude. The penalty paid for this efficiency is the need to update the index as the data

set changes. Except for data sets with critical hourly updates (such as stock quotes), this is generally not a problem. An enhancement to the indexing program to allow easier updating is given in section 14.7.4.

The description of the search process does not include the interface issues or the actual data retrieval issues. It is assumed that a natural language query is passed to the search process in some manner, and that the list of ranked record id numbers that is returned by the search process is used as input to some routine which maps these ids onto data locations and displays a list of titles or short data descriptors for user selection.

14.6.1 The Creation of an Inverted File

The inverted file described here is a modification to the inverted files described in Chapter 3 on that subject. It would be feasible to use structures other than simple inverted files, such as the more complex structures mentioned in that chapter, as long as the elements needed for ranking are provided. The use of a ranking system instead of a Boolean retrieval system has several important implications for supporting inverted file structures.

1. The use of ranking means that there is little need for the adjacency operations or field restrictions necessary in Boolean. Therefore, only the record id has to be stored as the location for each word, creating a much smaller index than for Boolean systems (in the order of 10% to 15% of the text size). If it is determined that the ranking system must also handle adjacency or field restrictions, then either the index must record the additional location information (field location, word position within record, and so on) as described for Boolean inverted files, or an alternative method (see section 14.8.4) can be used that does not increase storage but increases response time when using these particular operations. The inverted file presented here will assume that only record location is necessary.
2. The use of ranking means that strategies needed in Boolean systems to increase precision are not only unnecessary but should be discarded in favor of strategies that increase recall at the expense of precision. In the area of parsing, this may mean relaxing the rules about hyphenation to create indexing both in hyphenated and nonhyphenated form. In the area of stoplists, it may mean a less restrictive stoplist. For example, in a data set about computers, the ultra-high frequency term "computer" may be in a stoplist for Boolean systems but would not need to be considered a common word for ranking systems. In the area of stemming, a ranking system seems to work better by automatically expanding the query using stemming (Frakes 1984; Harman and Candela 1990) rather than by forcing the user to ask for expansion by wild-cards. A more appropri-